Mining Patterns with a Balanced Interval

Edgar H. de Graaf, Joost N. Kok and Walter A. Kosters

Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands
edegraaf@liacs.nl

Abstract. In many applications it will be useful to know those patterns that occur with a balanced interval, e.g., a certain combination of phone numbers are called almost every Friday or a group of products are sold a lot on Tuesday and Thursday.
In previous work we proposed a new measure of support (the number of occurrences of a pattern in a dataset), where we count the number of times a pattern occurs (nearly) in the middle between two other occurrences. If the number of non-occurrences between two occurrences of a pattern stays almost the same then we call the pattern balanced.
It was noticed that some very frequent patterns obviously also occur with a balanced interval, meaning in every transaction. However more interesting patterns might occur, e.g., every three transactions. Here we discuss a solution using standard deviation and average. Furthermore we propose a simpler approach for pruning patterns with a balanced interval, making estimating the pruning threshold more intuitive.

1 Introduction

Mining frequent patterns is an important area of data mining where we discover substructures that occur often in (semi-)structured data. In this work we will further investigate one of the simplest structures: itemsets. However the principles of balanced patterns are easily extended to sequential pattern mining, tree and graph mining. In earlier work we proposed an algorithm that discovers *stable patterns* that occur at regular moments, or rather in regular intervals, enabling us to mine for events that occur, e.g., every Friday. In this work we will introduce a new approach to mining for patterns with a stable interval. Note that the transactions in this paper have an order. In order to distinguish it from stable patterns we will call these new patterns *balanced patterns*. With this new approach we will offer solutions for problems in our work done in [5]:

- Patterns occurring in every transaction made it hard to discover patterns with a more interesting intermediate interval.
- The threshold for pruning was a certain value that a measure for stability needed to achieve. Even though a formula was given to estimate this value, an easily understandable value was lacking.

In Section 2.1 we will repeat some important definitions to make this work self contained, however in depth information can be found in [5].

We will define our approach to mining balanced patterns and show its usefulness. To this end, this paper makes the following contributions:

— We will **define balanced patterns and show their use**. These balanced patterns will enable the user to better filter uninteresting patterns (Section 2).

— Furthermore we will **propose an algorithm** that will enable us to mine balanced patterns (Section 3).

— Finally we will empirically show that **the algorithm can find interesting patterns** efficiently (Section 4).

A typical example is the mining of an access log from the Computer Science department of Leiden University. This access log will first be converted to sets of properties we are interested in, e.g., pages visited every half-hour. From here on we call this dataset the *website* dataset.

This research is related to work done on the (re)definition of support, using time with patterns and the incorporation of distance measured by the number of transactions between pattern occurrences. The notion of support was first introduced by Agrawal et al. in [1] in 1993. Since then many new and faster algorithms where proposed. We make use of ECLAT, developed by Zaki et al. in [12]. Steinbach et al. in [10] generalized the notion of support providing a framework for different definitions of support in the future. Our work is also related to work described in [8] where association rules are mined that only occur within a certain time interval. Furthermore there is some minor relation with mining data streams as described in [2, 7, 11], in the sense that they use time to say something about the importance of a pattern.

Finally this work is related to some of our earlier work. Results from [6] indicated that the biological problem could profit from incorporating consecutiveness into frequent itemset mining, which was elaborated in [3]. In the case of stable patterns we also make use of the transactions and the distance between them. Secondly in [4] it was mentioned that support is just another measure of saying how good a pattern fits with the data. There we defined different variations of this measure, and stability can been seen as one such variation. Stable patterns and an algorithm to discover them are defined in [5].

2 Regular Occurrence

In this section we will repeat the definition of stable patterns to better understand the problems and the difference with the definition of balanced patterns. In particular, patterns that occur at regular intervals (e.g., at equidistant time stamps) will be called stable or balanced. In the case of *stable patterns*, in order to judge this property, we will determine how often events occur “in the middle” between two other events [5]. In the case of *balanced patterns* we prune patterns that do not have at least one frequent intermediate distance (between all occurrences) and we filter those patterns that have a too high deviation for

all distances between successive occurrences. Furthermore we filter patterns that do not reach a certain minimal average distance for all successive occurrences.

2.1 Stable Patterns

In this paper a dataset consists of transactions that take zero time. Each transaction is an itemset, i.e., a subset of $\{1, 2, 3, \ldots, max\}$ for some fixed integer max. The transactions can have time stamps; if so, we assume that the transactions take place at different moments. We choose some notion of *distance* between transactions; examples include: (1) the distance is the time between the two transactions and (2) the distance is the number of transactions (in the *original* dataset) strictly in between the two transactions. In this paper we will use (2) in all our examples. We will define $\mathit{Trans}(p)$ as the series of transactions that contain pattern (i.e., itemset) p; the *support* of a pattern p is the number of elements in this ordered series.

We now define *w-stable patterns* as itemsets that occur frequent (support $\geq$ *minsup*) in the dataset and that have *stability value* $\geq$ *minstable*, where the values *minsup* and *minstable* are user defined thresholds. A *w-good triple* (L, M, R) consists of three transactions L, M and R, occurring in this order, such that $|\mathit{distance}(L, M) - \mathit{distance}(M, R)| \leq 2 \cdot w$; here w is a pregiven small constant ≥ 0, e.g., $w = 0$. The stability value of a pattern p is the number of w-good triples in $\mathit{Trans}(p)$, plus the number of transactions in $\mathit{Trans}(p)$ that occur as left endpoint in a w-good triple, plus the number of transactions in $\mathit{Trans}(p)$ that occur as right endpoint in a w-good triple.

Note that the stability value of a pattern p' with $p' \subseteq p$ is at least equal to that of p: the so-called Apriori or anti-monotone property. Also note that the stability value remains the same if we consider the dataset in reverse order.

In our work on stable patterns [5] we showed that equidistant events are "very" stable (in case $w = 0$).

Example 1. Suppose we have the following itemsets in our dataset:

transaction 1: $\{A, B, C\}$
transaction 2: $\{D, C\}$
transaction 3: $\{A, B, E\}$
transaction 4: $\{E, F\}$
transaction 5: $\{A, B, F\}$
transaction 6: $\{E, F\}$
transaction 7: $\{A, B, F\}$
transaction 8: $\{E, F\}$
transaction 9: $\{A, B, C\}$

The stability value (with $w = 0$) of $\{A,B\}$ is $4 + 3 + 3 = 10$, the maximal value possible. There are 4 0-good triples; we have 3 transactions that are left (right) endpoint of a 0-good triple (see picture below, left). If we insert two transactions $\{E, F\}$ between transaction 1 and 2, and also two between 8 and 9, we still have 4 0-good triples, but now we only have 2 transactions that are left (right)

endpoint of a good 0-triple (see picture below, right), leading to stability value $4+2+2=8<10$. This example shows that in order to guarantee equidistance one has to add left and right endpoints to the stability value.

2.2 Balanced Patterns

In this section we will define balanced patterns. We first discuss several problems and possibilities, and finally give the proper definition. We call the occurrences balanced if between two successive occurrences there is (almost) always the same amount of transactions.

The problem with patterns with balanced occurrences is that an itemset may occur less balanced than a superset of this itemset. Patterns occurring with a balanced interval do not have the *anti-monotone property*, where the subset is either equally good or better than the superset. In the balanced pattern case: the subset is not always more (or equally) balanced than the superset. This value will be used for pruning.

Example 2. Say that item A occurs in transactions 1, 4, 7 and 10 and item B occurs in transaction 4, 7, 10 and 13 then the itemset $\{A, B\}$ will occur in transaction 4, 7 and 10. Both A and B have three times two transactions between occurrences (successive and non-successive). However $\{A, B\}$ has only two times two transactions between occurrences because an occurrence can only become a non-occurrence and not the other way around.

For our definition of balanced patterns we first notice that all balanced occurrences (successive and non-successive) should have at least one intermediate distance a minimal number of times. Furthermore if you count the distances *between all occurrences* then this count is anti-monotone: a superset never has more of one particular distance. This is obvious because the number of occurrences will never increase for a superset and as a consequence the count of one particular distance will never increase. This property is also anti-monotone if we limit the distances we count, e.g., we count a distance only if it is smaller than 10 in-between transactions.

Example 3. The following table, where we only count upto 4 in-between transactions, is an example of counting the distances:

In-between Transactions (Distance)	Count
0	0
1	5
2	200
3	30
4	199

The *balanced value* for the pattern with these counts will be 200, the highest count in the table.

Still if we only look at the distance count we will not find the balanced patterns we want, since patterns that occur with very unbalanced intervals might still have a minimum amount of one particular distance. We filter those patterns by keeping the distance between occurrences that immediately succeed each other (instead of taking all distances). If a pattern is balanced then these distances should approach the average of all these distances. Their standard deviation will be near 0, since one distance should occur the most. Note that in calculating the standard deviation we do not limit the distances we consider. This can be done because the number of possible distances is far less for successive occurrences.

Now we can find all balanced patterns, however we will still find many patterns that are occurring every transaction. Their distance is almost always 0 and although they are well balanced they are often not interesting. These patterns can be filtered if we demand a certain average distance, e.g., if the user-defined threshold *minavg* is set to 1 then all these patterns will be filtered out, since their average distance approaches 0.

The definition of balanced patterns should be the following: A pattern is called a *balanced pattern* if among all occurrence pairs there is a distance that occurs at least a user-defined number of times (*minnumber*) and the distance between successive occurrences have maximally a user-defined standard deviation (*maxstdev*) and minimally a user-defined average (*minavg*).

3 Algorithm

We now consider algorithms that find all frequent itemsets, given a database. A *frequent* itemset is an itemset with support at least equal to some pre-given threshold, the so-called *minsup*. Thanks to the Apriori property many efficient algorithms exist. However, the really fast ones rely upon the concept of FP-tree or something similar, which does not keep track of in-between distances. This makes these algorithms hard to adapt for use in balanced patterns.

One fast algorithm that does not make use of FP-trees is called Eclat [12]. Eclat grows patterns recursively while remembering which transactions contained the pattern, making it very suitable for balanced patterns. In the next recursive step only these transactions are considered when counting the occurrence of a pattern. All counting is done by using a matrix and patterns are extended with new items using the order in the matrix. This can easily be adapted to incorporate balance counting.

Our algorithm BalanceClat will use the Eclat algorithm. However instead of counting support we count the different distances between all occurrences, e.g., pattern A has 10 times 3 transactions between occurrences. We will prune on this value instead of pruning on the minimal support threshold. In

this case the user-defined threshold will be the minimal number of times at least one of $\ell + 1$ distances $\{0, 1, 2, \ldots, \ell\}$ is seen. For balanced patterns we consider this threshold to be the *minnumber* threshold. As said before, we can only find balanced patterns if we also demand a maximal standard deviation for distances between occurrences. This will be done by introducing the *maxstdev* threshold. Finally we are not interested in patterns occurring in every transaction. We introduce a third user-defined threshold that demands a minimal average distance: *minavg*. For *maxstdev* and *minavg* we only use distances between successive occurrences and for *minnumber* all distances $\leq \ell$.

We now propose a more general definition. Suppose we have an itemset I and let $O_j \in \{0, 1\}$ $(j = 1, 2, \ldots, r)$ denote whether or not the j^{th} transaction in some subset $\mathcal{S}$ of the database $\mathcal{D}$ contains I (O_j is 1 if it does contain I, and 0 otherwise; the O's are referred to as the *O-series*), $r = |\mathcal{S}|$. The function $\varphi : N \to N$ is a translation from the index j for the j-th transaction in $\mathcal{S}$ to the index k giving the position of the same transaction in $\mathcal{D}$.

The main adaptation to ECLAT is replacing support with a *balance value* denoted with t. Also it calculates the standard deviation (*stdev*) and average distance (*avgdist*) for the successive occurrences:

$j := 2$, $h := -1$
succdists := sequence of distance counts between *successive* occurrences
alldists := sequence of distance ($\leq \ell$) counts between *all* occurrences
while ($j \leq r$) **do**
  if ($O_j = 1$) **then**
    $i := 1$
    while ($i < j$) **do**
      if ($O_i = 1$ **and** $\varphi(j) - \varphi(i) - 1 \leq \ell$) **then**
        $alldists_{\varphi(j)-\varphi(i)-1} := alldists_{\varphi(j)-\varphi(i)-1} + 1$
      fi
      $i := i + 1$
    od
    if ($h \neq -1$) **then**
      $succdists_{\varphi(j)-\varphi(h)-1} := succdists_{\varphi(j)-\varphi(h)-1} + 1$
    fi
    $h := j$
  fi
  $j := j + 1$
od
$t := max(alldists)$, the largest count in the sequence
stdev := standard deviation for *succdists*
avgdist := average for *succdists*, also denoted with $avg(succdists)$

The standard deviation for *succdists* can simply be calculated in the following way:

$$\sqrt{\sum_i (avg(succdists) - i)^2 \cdot succdists_i \ / \ \sum_i succdists_i} \qquad (1)$$

Eclat can now prune using the balance value t (if $t < minnumber$) and patterns are only displayed if their standard deviation and average distance are sufficient. These are straightforward adaptations that will not be given in detail.

Standard deviation changes if patterns occur less balanced in a certain small number of successive transactions, small periods. In some cases it might be preferable to remove the influence of these periods. One possible approach is to calculate average distance and the standard deviation for *frequent distances* (for successive occurrence) only. The value for filtering with standard deviation for the sequence $\mathcal{Q} = \langle y | y = succdist_i, y \geq mindistfreq \rangle$ will be:

$$stdev = \begin{cases} \sqrt{\sum_i (avg(\mathcal{Q}) - i)^2 \cdot \mathcal{Q}_i \ / \ \sum_i \mathcal{Q}_i} & \text{if } \mathcal{Q} \text{ is } \textit{not} \text{ empty} \\ maxstdev + 1 & \text{otherwise} \end{cases} \qquad (2)$$

Note that via the threshold *mindistfreq* the user decides when a distance is considered frequent.

4 Results and Performance

The experiments were done for three main reasons. First of all we want to show *known balanced patterns will be found* also in the case of noise. Secondly we want to show that *interesting balanced patterns can be found* in real datasets. Finally we want to *show runtime for real data* and *how the minnumber threshold influences runtime.*

Our implementation of the balanced pattern mining algorithm is called BalanceClat. All experiments were performed on an Intel Pentium 4 64-bits 3.2 GHz machine with 3 GB memory. As operating system Debian Linux 64-bits was used with kernel 2.6.8-12-em64t-p4.

The synthetic datasets used in our first experiment are called *find-noise-x%* where x is a noise value ranging from 0 to 30. E.g., if the noise is 10%, this means there is a 10% chance that one element of the balanced pattern does not occur when it should. In each of these *find-noise-x%* datasets one pattern of 5 of the 200 items occur every 4 transactions (so distance = 3) and each dataset has 2,000 transactions. If 5 items always occur balanced like this, we expect to find $\sum_{k=1}^{5} 5!/(5-k)!k! = 31$ patterns. First the BalanceClat algorithm is executed with $maxstdev = 2.5$, $minavg = 2.0$ and $minnumber = 150$. Figure 1 displays the number of expected patterns that were found by the algorithm. We see that the algorithm detects most patterns up to a noise level of 15%. Due to the way we generate noise, long patterns become less likely as the noise level increases. With a high noise level we only find the patterns of 1 item in length. This can be improved if we change our settings for *maxstdev* and *minavg*, but we kept them fixed for comparison reasons.

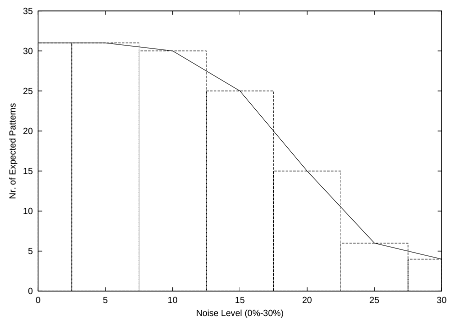

Fig. 1. The effect of noise on the algorithm.

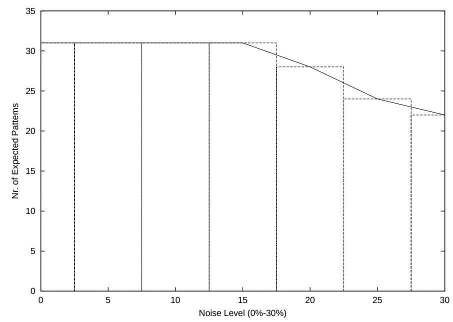

Fig. 2. The effect of noise on the algorithm, *mindistfreq* $= 50$.

We can use the *mindistfreq* threshold to decrease the influence of small noisy periods on the balanced occurrences. Figure 2 shows how the effect of noise becomes less if we set a *mindistfreq* of 50. Now one also finds more of the other patterns that happen to occur reasonably balanced, however we can filter them by lowering *maxstdev*.

With our next experiment we want to show the effect of dataset size on the algorithm, scalability. In Figure 3 first the runtime drops; this is because many patterns have distances occurring only a few times. E.g., when the dataset size is 100 then $minnumber = 0.1 \cdot 100 = 10$. Many patterns have distances that occur at least 10 times. As this effect becomes less, runtime increases and eventually it becomes nearly linear.

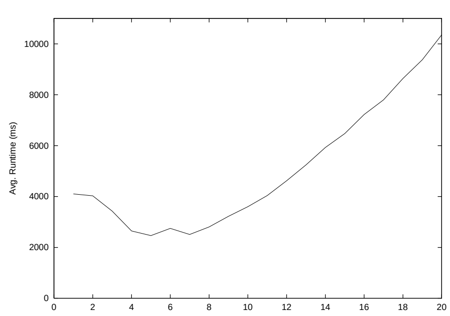

Fig. 3. Runtime in ms for different dataset sizes; *minnumber* is 10% of the dataset size ($maxstdev = 1.0$, $minavg = 2.0$, $\ell = 10$).

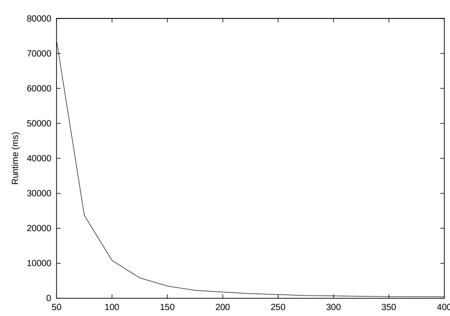

Fig. 4. Runtime in ms for different values of *minnumber* ($maxstdev = 1.0$, $minavg = 2.0$, $\ell = 10$).

The BalanceClat algorithm was also tested on the website dataset. This dataset is based on an access log of the website of the Computer Science de-

partment of Leiden University, as said before. It contains all 1,991 items of the web-pages that were visited, grouped in half-hour blocks, so each of the 1,488 transactions contains the pages visited during one half-hour. Figure 4 shows how the runtime for the website dataset drops fast as *minnumber* increases.

Table 1 shows the count for distances between successive occurrences. It shows that this particular pattern, consisting of the websites of two professors of the same group and the main page, occurs often with a successive distance of 0, 1 or 2. This pattern probably is caused by students having courses from both professors and some of these students access both pages nearly every half an hour.

In-between Transactions (Distance)	Count
0	385
1	171
2	78
3	25
4	23

Table 1. The distances (with count ≥ 20) between successive occurrences and their counts for one pattern (two professors & the main page) in the website dataset ($maxstdev = 2.0$, $minavg = 1.0$, $\ell = 10$).

Finally we also applied the BalanceClat algorithm to the Nakao dataset used in [3]. In this dataset each of the 2,124 transactions is a clone located on the human chromosomes. The items are the numbers of patients with a higher than normal value for this clone (≥ 0.225). The specifics of the dataset can be found in [9]. The parameter *minavg* was set 0.0, because the interesting patterns are expected to occur very close to each other. Also $mindistfreq = 10$ because patterns where expected to have small periods of transactions where they occurred unbalanced. Furthermore $maxstdev = 0.2$, $\ell = 10$ and $minnumber = 100$. Results where similar to results found with consecutive support as presented in [3] where most consecutive patterns occurred close together in chromosome 9. In the future we plan to investigate this futher.

5 Conclusions and Future Work

We have presented a new way of mining for patterns occurring with a regular interval. In comparison with our previous method we now use a pruning threshold *minnumber* that is more intuitive to users. With it the user only indicates the number of times at least one intermediate distance should occur. Such a distance is the number of transactions between two occurrences of the pattern (we consider only distances below a maximal distance).

In this work we call patterns with a regular interval balanced and we discuss an algorithm to find them efficiently. Its runtime performance and scalability is evaluated through experimentation.

Finally in the future we plan to use balanced patterns in combination with new ways of filtering to facilitate the discovery of new patterns further. Also research will be done on effectively visualizing balanced patterns.

Acknowledgments This research is carried out within the Netherlands Organization for Scientific Research (NWO) MISTA Project (grant no. 612.066.304).

References

1. Agrawal, R., Imielinski, T., Srikant, R.: *Mining Association Rules between Sets of Items in Large Databases*. In Proc. ACM SIGMOD Conference on Management of Data (1993), pp. 207–216.
2. Chen, Y., Dong G., Han J., Wah, B., Wang J.: *Multidimensional Regression Analysis of Time-series Data Streams*. In Proc. 28th Int. Conference on Very Large Data Bases (VLDB 2002), pp. 323–334.
3. Graaf, E.H. de, Graaf, J.M. de, Kosters, W.A.: *Using Consecutive Support for Genomic Profiling*. In Proc. of the ECML/PKDD Workshop on Data and Text Mining for Intergrative Biology (BIOWS 2006), pp. 16–27.
4. Graaf, E.H. de, Kosters, W.A.: *Efficient Feature Detection for Sequence Classification in a Receptor Database*. In Proc. Seventeenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2005), pp. 81–88.
5. Graaf, E.H. de, Kosters, W.A.: *Mining for Stable Patterns: Regular Intervals Between Occurrences*. In Proc. Seventeenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2006), pp. 149–155.
6. Graaf, J.M. de, Menezes, R.X. de, Boer, J.M., Kosters, W.A.: *Frequent Itemsets for Genomic Profiling*. In Proc. 1st International Symposium on Computational Life Sciences (CompLife 2005), LNCS 3695, Springer, pp. 104–116.
7. Giannella, C., Han, J., Pei J., Yan, X., Yu, P.: *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. In Proceedings of the NSF Workshop on Next Generation Data Mining (NGDM 2002), pp.191–210.
8. Li, Y., Ning, P., Wang, X.S., Jajodia, S.: *Discovering Calendar-based Temporal Association Rules*. In Proc. of the 8th Int Symposium on Temporal Representation and Reasoning (TIME 2001), pp. 111–118.
9. Nakao, K., Mehta, K.R., Fridlyand, J., Moore, D.H., Jain, A.N., Lafuente, A., Wiencke, J.W., Terdiman, J.P., Waldman, F.M.: *High-resolution Analysis of DNA Copy Number Alterations in Cororectal Cancer by Array-based Comparative Genomic Hybridization*. In Carcinogenesis 25 (2004), pp. 1345–1357
10. Steinbach, M., Tan, P., Xiong, H., Kumar, V.: *Generalizing the Notion of Support*. In Proc. 10th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2004), pp. 689–694.
11. Teng, W., Chen, M., Yu, P.S.: *A Regression-based Temporal Pattern Mining Scheme for Data Streams*. In Proc. 29th Int. Conference on Very Large Data Bases (VLDB 2003), pp. 93–104.
12. Zaki, M., Parthasarathy, S., Ogihara, M., Li, W.: *New Algorithms for Fast Discovery of Association Rules*. In Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD 1997), pp. 283–296.